

SAS Macro Design Issues

Ian Whitlock
Westat, An Employee-Owned
Research Corporation
ianwhitlock@westat.com

The Questions

- ↳ What makes macro code difficult to read?
- ↳ How can design help?
- ↳ What makes a good design good?
- ↳ Are there any basic principles?

2

The Problem

- ↳ Readability
- ↳ Programs must communicate
 - ▶ To machine
 - ▶ And humans

3

First Tools of Communication

- ↳ Names
- ↳ Parameters
- ↳ Values

4

Variables

```
%let data = lib.mydata ;
```

- What does the name data suggest?
 - ▣ SAS data or external data?
 - ▣ Input data or output data?
 - ▣ Will any good SAS programmer know?
- What about DSN, File, IN, or INPUT?

5

Parameters

```
%let data = lib.mydata ;
```

```
libname lib "c:\myproject\dat" ;  
  
title3 "The data set is &data" ;  
proc print data = &data ;  
run ;
```

6

Values

- ↳ Values are also important
- ↳ Not all values are good values

Macro variable values are code pieces

- ▶ The names must communicate
- ▶ The pieces must communicate

7

Values

Problem: Convert
`cdt = "APR 15 2002"`
to a SAS date

Consider
`date = input (cdt, date9.);`
But CDT in wrong order

8

Values

Expression:

```
scan(cdt, 2) || scan(cdt, 1) || scan(cdt, 3)
```

Consider:

```
%let x = scan(cdt, ;  
date=input (&x 2) ||&x 1) ||&x 3));
```

9

Values

↳ SAS syntax is strong constraint on code

- ▶ Predictable
- ▶ Harder to make unreadable

↳ Macro provides weaker constraint

- ▶ Less predictable
- ▶ Easy to make unreadable

10

Readability

↳ The front line weapons

- ▶ Variable names must convey intent
- ▶ Variable values must be chosen to reflect structure

11

Parameters versus Variables

↳ Parameters defined external to program

↳ Parameters drive the program

↳ Variables defined internally to create flexibility

12

Macro Definition

- ↳ Starts with %MACRO statement
- ↳ Ends with %MEND statement
- ↳ Macro compile time
 - No macro instructions are executed
 - Macro instructions are compiled

13

Macro Definition

```
%macro tprint ( data = &syslast  
                ) ;  
    %let data = &data ;  
    title3 "Data = &data" ;  
    proc print data = &data ;  
    run ;  
    title3 ;  
%mend tprint ;
```

14

Macro Invocation

- ↳ Macro execution time
- ```
%tprint (data = lib.mydata)
%tprint (data = temp)
%tprint (data = results)
```

15

## Features

- ↳ Macro name helps to explain function
- ↳ Data parameter acts like SAS default
- ↳ Function is well defined
- ↳ Extendable with more parameters

16

## Macro Definition

```
%macro tprint (data=&syslast,
 obs=20, tl=3);
 %let data = &data ;
 title&tl "Data=&data(obs=&obs)";
 proc print data=&data(obs=&obs);
 run ;
 title&tl ;
%mend tprint ;
```

17

## Parameters

- ↳ The role of parameters cannot be over emphasized in looking for good design
  - ▶ A macro promises to provide some service
  - ▶ The parameters provide the contract the user must fulfill to obtain the service

18

## Macro Decisions

- ↳ Which code to generate
- ↳ Which macro instruction to execute

```
%if condition %then consequent ;
```

19

## TPRINT Again

```
%macro tprint (data=&syslast,
 obs=20, tl=3);
 %global debug; /* Y or N */
 %if %upcase(&debug) = Y %then
 %do ;
 title&tl
 "Data=&data(obs=&obs)";
```

20

## TPRINT Again

```
proc print
 data=&data(obs=&obs) ;
run ;
 title&tl ;
%end ;
%mend tprint ;
```

21

## Rearrange Date

```
%macro rearrange (v = cdt) ;
 scan(&v, 2) || scan(&v, 1) || scan(&v, 3)
%mend rearrange ;

date = input(%rearrange(v=cdt),
 date9.) ;
```

22

## Date Conversion

```
%macro convdate (v) ;
 input (
 scan(&v, 2) || scan(&v, 1) || scan(&v, 3),
 date9.) /* no semi colon! */
%mend convdate ;

date = %convdate(cdt) ;
```

23

## Too Many %IFs

```
%if %upcase(&recordset) = "RNDJAN" %then %do;
 if month(datepart(rnddate)) = 01 ;
%end; %else
%if %upcase(&recordset) = "RNDFEB" %then %do;
 if month(datepart(rnddate)) = 02 ;
%end; %else
%if %upcase(&recordset) = "RNDMAR" %then %do;
 if month(datepart(rnddate)) = 03 ;
%end; %else ...
```

24

## Too Many %IFs

```
%macro subset (site = 11,
 var = rnddate ,
 month = Jan) ;
data out2 ;
 set efr.out1(where=(center=&site));
 if upcase(put(datepart(&var),
 monname3.)) = %upcase(&month);
run ;
%mend subset ;
```

25

## Too Many %IFs

```
%macro subset (site = 11 ,
 var = rnddate ,
 month = Jan) ;
 where center=&site
 and upcase(put(datepart(&var),
 monname3.)) = %upcase(&month);
%mend subset ;
```

26

## Too Many %IFs

```
data out2 ;
 set efr.out1 ;
 %subset (site = 11 ,
 var = rnddate ,
 month = Jan)
run ;
```

27

## The Secret of Lists

↳ A lot of SAS code involves lists

- ▶ List of variables
- ▶ List of data sets
- ▶ list of formats
- ▶ list of assignment statements

↳ List manipulation plays a central role in many problems with macro solutions

28

## The Secret of Lists

↳ SQL is a super list maker

```
proc sql noprint ;
 select name
 into :varlist separated by " "
 from dictionary.columns
 where libname = "WORK"
 and memname = "MYDATA"
 ;
quit ;
```

29

## The Secret of Lists

↳ SQL is a super array maker

```
proc sql noprint ;
 select name into :var1-var9999999
 from dictionary.columns
 where libname = "WORK"
 and memname = "MYDATA" ;
 %let nv = &sqllobs ;
quit ;
```

30

## The Secret of Lists

↳ Lists and arrays are interchangeable

- ▶ Every element of a list can be an element in an array
- ▶ Every element of an array can be concatenated to make a list

31

## Macro Looping

```
%macro MkList (v1=a, v2=b, v3=c);
 %local i ;
 %global list ; /* return value */
 %let list = ;
 %do i = 1 %to 3 ;
 %let list = &list &&v&i ;
 %end ;
%mend MkList ;
```

32

## Multiple Ampersands

&i=1: &&v&i --> &v1 --> a

↳ The Rules

- ▶ Scan left to right
- ▶ Change two ampersands to one
- ▶ Resolve ampersand text
- ▶ Copy text
- ▶ Repeat until no ampersands

33

## Macro Looping

```
%macro MkArray (list=a b c,
 root=v) ;
 %local i /* looping index */
 w; /* word in list */
 %let i = 1 ;
 %let w = %scan (&list, &i);
```

34

## Macro Looping

```
%do %while (%length(&w) > 0) ;
 %global &root&i ;
 %let &root&i = &w ;
 %let i = %eval (&i + 1) ;
 %let w = %scan(&list, &i);
%end ;
%mend MkArray ;
```

35

## CALL EXECUTE

↳ Data step function sends a character string to the macro facility for immediate execution

↳ Macro generates code

- ▶ Code is generated immediately
- ▶ Code is executed after generating step ends

36

## CALL EXECUTE

```
data _null_ ;
 input data $char41. ;
 call execute (
 '%tprint (data=' || trim(data) || ")"
) ;
cards ;
lib.mydata
work.yourdata
;
```

37

## Design

↳ Consider example significant enough to show principles

38

## Big Example

- ↳ 80 data sets with ID and some variables
- ↳ Some IDs need some variables blanked
- ↳ Data set IDCNTL (ID P1-P40)
- ↳ Data set CNTL (Key Mem Type Vars)
- ↳ When  $P_i = 1$  blank set of vars in member (see CNTL)

39

## Big Example

| IDCNTL |    |    |    |
|--------|----|----|----|
| ID     | P1 | P2 | P3 |
| 1      | 1  | 0  | 0  |
| 2      | 0  | 1  | 1  |
| ...    |    |    |    |

| CNTL |     |      |     |
|------|-----|------|-----|
| KEY  | MEM | TYPE | VAR |
| P1   | DS1 | CHAR | Z   |
| P1   | DS1 | NUM  | X Y |
| ...  |     |      |     |

40

## Plan

- ↳ %BlankLib
  - Get array of keys from control file
  - Loop through array calling %BlankKey
- ↳ %BlankKey
  - For given key use CNTLIN option to make a format of IDs needing blanking
  - Use CALL EXECUTE to loop through relevant members calling %BlankMem

41

## Plan

- ↳ %BlankMem
  - Data step to modify data set
  - Use %GenAssign to blank char vars
  - Use %GenAssign to blank num vars
- ↳ %GenAssign
  - Loop through list of vars making blank assignments

42

## Features Illustrated

### ↳ Good design

- Each macro has well determined function
- Clear parameter interface to determine responsibilities

### ↳ Patterns of macro programming

- Looping through array of macro vars
- CALL EXECUTE to execute a macro with parameters determined by data
- Looping over a list

43

## Big Example

```
/* id control information */
data idcntl ;
 input id $ p1 p2 ;
cards ;
1 1 .
2 . 1
;
```

44

## Big Example

```
/* variable control information */
data cntl (keep = key mem type vars);
 input key $ mem : $char3.
 type : $char4. vars &: $char1000. ;
cards ;
p1 ds1 CHAR z
p1 ds1 NUM x y
p1 ds2 CHAR a
p1 ds2 NUM y
p2 ds2 NUM y z
;
```

45

## Big Example

```
/* test data */
data ds1 (keep = a x y z)
 ds2 (keep = a y z) ;
 input id $ @@ ;
 retain x y 5 a z "emp";
cards ;
1 2 3
;
```

46

## BlankLib

```
%macro blanklib (lib = , cntl = cntl) ;
%local i nk ;
/* get list of keys */
proc sql noprint ;
 select distinct key into :key1- :key99
 from cntl ;
%let nk = &sqllobs ;
quit ;
```

47

## BlankLib

```
%do i = 1 %to &nk ;
 %blankkey (key = &&key&i, lib = &lib)
%end ;
%mend blanklib ;
```

48

## BlankKey

```
%macro blankkey (key= , lib =) ;
 %let key = %upcase(&key) ;
```

49

## BlankKey

```
/* make format */
data fmdata ;
 retain fmtname "$&key.W"
 label "W" ;
if eof then do ;
 HLO = "0" ; label = "N" ;
 output ;
end ;
```

50

## BlankKey

```
set idcntl (keep = id &key
 rename = (id=start)
 where = (&key = 1)
) end = eof ;
output ;
run ;
proc format cntlin=fmdata fmtlib ;
run ;
```

51

## BlankKey

```
/* call blankmem for each member */
data _null_ ;
length charlist numlist $ 32000 ;
retain charlist ;
set cntl
 (where = (upcase(key)="&key"));
by mem type ;
if type="CHAR" then charlist=vars ;
if type="NUM" then numlist= vars ;
```

52

## BlankKey

```
if last.mem ;
call execute (
 '%blankmem ('
 || "lib=&lib, key=&key"
 || ", mem=" || trim(mem)
 || ", charlist=" || trim(charlist)
 || ", numlist =" || trim(numlist)
 || ")"
) ;
```

53

## BlankKey

```
charlist = " " ;
run ;
%mend blankkey ;
```

54

## BlankMem

```
%macro blankmem (lib=, key=, mem=,
 charlist=, numlist=);

 data &lib..&mem ;
 modify &lib..&mem ;
 if put (id, S&key.W.)="W"
 then
```

55

## BlankMem

```
do ;
 %GenAssign(list=&charlist,
 val="#")
 %GenAssign(list=&numlist,
 val=.A)
 replace ;
end ;
run ;
%mend blankmem ;
```

56

## GenAssign

```
%macro GenAssign(list=, val=) ;
 %local i w ;
 %let i = 1 ;
 %let w = %scan(&list,&i,%str()) ;
 %do %while (%length(&w) > 0) ;
 &w = &val ;
 %let i = %eval (&i + 1) ;
 %let w=%scan(&numlist,&i,%str()) ;
 %end ;
%mend GenAssign ;
```

57

## Conclusion

### ↳ Clarity

- ▶ Names show intent
- ▶ Parameters show contract
- ▶ Well determined function of macro

### ↳ Design

- ▶ Minimize %IF
- ▶ Emphasized list manipulation

58

## Notes

Westat Disclaimer:

The contents of this paper are the work of the author(s) and do not necessarily represent the opinions, recommendations, or practices of Westat.

SAS is a registered trademark the SAS Intsitute Inc in the USA and other countries.

® indicates USA Registration.

59